Summary and Review of: Gate-Level Netlist Reverse Engineering for Hardware Security: Control Logic Register Identification

Brian Weber

1 Abstract

It is easy and relatively inexpensive to obtain a full netlist from fabricated chips using reverse engineering, however there is still a lack of tools to analyze this netlist to look for malicious functionality. This paper introduces a solution for analyzing and categorizing the funtionality of registers given a netlist called RELIC (Reverse Engineering Logic Identification Classification). Using the methods described in the paper control logic registers were able to be distinguished from data registers in circuits of varying complexity.

2 Methods

Given an arbitrary netlist, RELIC takes an arbitrary netlist and produces assigns scores called "Similarity Scores" to register gate pairs by using Dynamic Programming and graph algorithms. Based on these scores, the registers in a netlist are more easily classified, and it is easier to distinguish between real and malicious registers. RELIC assumes register pairs from the same datapath have similar deep logical structures. Using this assumption, RELIC analyzes and compares the fan-in structure of each register to generate similarity scores. Using this technique, logic registers are found by finding registers that are not part of a data path. In addition, trojan logic registers are identified based on the assumption that it is unlikely they will have similar logic to valid logic registers.

To generate these scores, RELIC first reduces all input structures to AND-OR-INV logic (converting xor to 2 ands and an or, nand to and+inv etc.). To solve obfuscation (accidental or intentional), logic gates are merged until they cannot be merged anymore. No gates are merged that are used for registers. After preprocessing, similarity scores are generated based on the topological structure of the netlist. Scores are generated for arbitrary pairs of logic vertexes from 0 to 1. A score of 0 means the fan-in structures of the 2 registers have no similarities, while a score of 1 means they are identical. If the score exceeds some predetermined threshhold it is added to a bipartate graph. memoization, results are cached to prevent recomputation of 2 vertexes with the same value. This process is done recursively up to a predtermined depth. After the scores are calculated, registers are classified. Registers that are similar to many other registers are typically not logic affecting registers.

3 Results

RELIC seems to be very good at classifying registers, and based on the tests done, did so efficiently. At a minimum, it correctly classified 89.1% of registers. It was also able to classify registers in a circuit with 12576 gates with 3968 registers with 100% accuracy in 4 minutes. Circuits that have higher logical complexity showed that they needed lower depths to be computed efficiently. This in turn negatively affected the accuracy of the computation, causing the similarity scores to be higher than they should have.

4 Discussion

This seems to be a strong option for detecting malicious registers given a netlist. Combining this with the controllability and observability analysis described in the previous provides a robust set of tools for detecting malicious structure in netlists, though I am not completely sure what this method provides that the controllability and observability analysis does not provide. This method seems to be very efficient for circuits of low logic complexity, but as the MC8051 example shows, for circuits with higher logical complexity, it becomes more difficult to compute the similarity scores. This makes me worry about how this method scales to large and logically complex circuits. In addition, I would have liked to see more tests done on many more and larger circuits. Finally, while accuracy was mostly good, it was inconsistent depending on the structure of the design, which may mean this inconsitency might be able to be abused by an adversary.